2

Ants Constructing Rule-Based Classifiers

David Martens[1], Manu De Backer[1], Raf Haesen[1],
Bart Baesens[2,1], Tom Holvoet[3]

[1] Department of Applied Economic Sciences, K.U.Leuven
Naamsestraat 69, B-3000 Leuven, Belgium
{David.Martens;Manu.Debacker;Raf.Haesen;Bart.Baesens}@econ.kuleuven.be
[2] University of Southampton, School of Management
Highfield Southampton, SO17 1BJ, United Kingdom
Bart@soton.ac.uk
[3] Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium
Tom.Holvoet@cs.kuleuven.be

Summary. This chapter introduces a new algorithm for classification, named AntMiner+, based on an artificial ant system with inherent self-organizing capabilities. The usage of ant systems generates scalable data mining solutions that are easily distributed and robust to failure. The introduced approach differs from the previously proposed AntMiner classification technique in three aspects. Firstly, AntMiner+ uses a $\mathcal{MAX}$-$\mathcal{MIN}$ ant system which is an improved version of the originally proposed ant system, yielding better performing classifiers. Secondly, the complexity of the environment in which the ants operate has substantially decreased. This simplification results in more effective decision making by the ants. Finally, by making a distinction between ordinal and nominal variables, AntMiner+ is able to include intervals in the rules which leads to fewer and better performing rules. The conducted experiments benchmark AntMiner+ with several state-of-the-art classification techniques on a variety of datasets. It is concluded that AntMiner+ builds accurate, comprehensible classifiers that outperform C4.5 inferred classifiers and are competitive with the included black-box techniques.

2.1 Introduction

In recent decades, innovative storage technologies and the success of the Internet have caused a true explosion of data. This data is typically distributed, continuously updated and contains valuable, yet hidden knowledge. Data mining is the overall process of extracting knowledge from this raw data. Although many techniques have been proposed and successfully implemented, few take into account the importance of the comprehensibility aspect of the generated models or the ability to deal with distributed data. Artificial ant systems are inspired on real ant colonies and are specifically designed to provide robust, scalable and

D. Martens et al.: *Ants Constructing Rule-Based Classifiers*, Studies in Computational Intelligence (SCI) **34**, 21–43 (2006)
www.springerlink.com

distributed solutions. By performing local actions and indirect communication only, ants are able to achieve complex overall behavior. The approach described in this chapter, named AntMiner+, takes advantage of the inherent benefits of ant systems and puts them in a data mining context. Comprehensible, accurate classifiers in the form of simple **if-then-else** rules are extracted from data by the ants. The environment of the ants is defined as a directed acyclic graph (DAG) where an ant, walking from start to end, gradually constructs a rule. AntMiner+ uses a $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, which is an improved version of the originally proposed ant system [41] and enhances the performance by a stronger exploitation of the best solutions.

The remainder of this chapter is structured as follows. In Sect. 2.2 we shortly explain the basics of ant systems, data mining and introduce the use of ant systems for data mining. This is further elaborated on in Section 2.3 where we explain the workings of our approach: AntMiner+. The final sections report on the results of our experiments on various datasets.

2.2 Ant Systems and Data Mining

2.2.1 Ant Systems

Artificial ant systems are inspired on the behavior of real ant colonies and are part of a relatively new concept in artificial intelligence, called swarm intelligence [5]. Swarm Intelligence is the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge. A biological ant is a simple insect with limited capabilities but an ant colony is able to behave in complex manners and come to intelligent solutions for problems such as the transportation of heavy items and finding the shortest path between the food source and the nest. This complex behavior emerges from self-organization and indirect communication between the ants. The indirect way of communication, through the environment rather than directly between the individuals, is also known as stigmergy [18]. More specifically, ants communicate through a chemical substance called pheromone that each ant drops on its path. When an ant finds a pheromone trail it is likely to follow this path and reinforce the pheromone. The pheromone trail intensity is increased and the path will become more likely to be followed by other ants. In turn, when no ants follow the same path the pheromone trail intensity decreases, this process is called evaporation.

The same ideas are used for artificial ant systems [11]: a number of computational concurrent and asynchronous agents move through their environment and by doing so incrementally construct a solution for the problem at hand. Ants move by applying a stochastic local decision policy based on two parameters, the pheromone and heuristic values. The pheromone amount of a trail is a measure for the number of ants that recently have passed the trail and the heuristic value is a problem dependent value. When an ant comes at a crossroad, it is more likely to choose the trail with the higher pheromone and heuristic values. When an ant arrives at its destination, the ant's solution is evaluated and the trail followed by the ant is updated according to its quality. Updating the trails entails two phenomena: evaporation and reinforcement. Evaporation means that the pheromone level of the trails are diminished gradually. In this way less accurate trails will disappear. Reinforcement means that the pheromone level is increased proportionally to the quality of the corresponding candidate solution for the target problem. As a result, the solution provided by the ants will converge to a (sub)optimal solution of the problem.

In essence, the design of an ant system implies the specification of the following aspects:

- An environment that represents the problem domain in such a way that it lends itself to incrementally building a solution for the problem;
- A problem dependent heuristic evaluation function (η), which represents a quality factor for the different solutions;
- A rule for pheromone updating (τ), which takes into account the evaporation and the reinforcement of the trails;
- A probabilistic transition rule based on the value of the heuristic function (η) and on the strength of the pheromone trail (τ) that is used to iteratively construct a solution;
- A clear specification of when the algorithm converges to a solution.

Ant systems have shown to be a viable method for tackling hard combinatorial optimization problems [10]. A short overview of the literature, though not exhaustive, is provided in Table 2.1.

The performance of traditional ant systems, however, is rather poor on larger problems [37]. Stützle et al. [41] advocate that improved performance can be obtained by a stronger exploitation of the best solutions, combined with an effective mechanism for avoiding early search stagnation[4]. The authors propose a $\mathcal{MAX}$-$\mathcal{MIN}$ ant system ($\mathcal{MM}$AS) that differs from a normal ant system in three aspects:

- After each iteration only the best ant is allowed to add pheromone to its trail. This allows for a better exploitation of the best solution found;
- To avoid stagnation of the search, the range of possible pheromone trails is limited to an interval $[\tau_{min}, \tau_{max}]$;

[4]The situation where all ants take the same path and thus describe the same solution.

- Each trail is initialized with a pheromone value of τ_{max}, as such the algorithm achieves a higher exploration at the beginning of the algorithm.

Table 2.1. Literature Overview

Overview of the applications of Ant Systems		
Data Mining	Clustering	Abraham et al. [1] Handle et al. [20] Schockaert et al. [33]
	Classification	Parpinelli et al. [28, 29] Liu et al. [22] Ramos et al. [31]
Operations Research	Traveling Salesman Problem	Dorigo et al. [12] Gambardella et al. [16, 17] Eyckelhof et al. [13] Stützle et al. [38, 39, 40]
	Vehicle Routing Problem	Wade et al. [45] Bullnheimer [6] Cicirello et al. [8]
	Quadratic Assignment Problem	Maniezzo et al. [24, 25] Gambardella et al. [15] Stützle et al. [36]
	Scheduling Problems	Colorni et al. [9] Socha et al. [35] Forsyth et al. [14]
	Telecommunications	Schoonderwoerd et al. [34] Di Caro et al. [7]

2.2.2 Data Mining

Over the past decades we have witnessed an explosion of data. Although much information is available in this data, it is hidden in the vast collection of raw data. Data mining entails the overall process of extracting knowledge from this data and addresses the rightly expressed concern of Naisbitt [27]:

"We are drowning in information but starving for knowledge."
– John Naisbitt

Different types of data mining are discussed in the literature [2], such as regression, classification and clustering. The task of interest here is classification, which is the task of assigning a datapoint to a predefined class or group according to its predictive characteristics. The classification problem and accompanying data mining techniques are relevant in a wide variety of domains such as financial engineering, medical diagnostic and marketing. The result of a classification technique is a model which makes it possible to

classify future data points based on a set of specific characteristics in an automated way. In the literature, there is a myriad of different techniques proposed for this classification task, some of the most commonly used being C4.5, logistic regression, linear and quadratic discriminant analysis, k-nearest neighbor, artificial neural networks and support vector machines [19].

The performance of the classifier is typically determined by its accuracy on an independent test set. Benchmarking studies [3] have shown that the non-linear classifiers generated by neural networks and support vector machines score best on this performance measure. However, comprehensibility can be a key requirement as well, demanding that the user can understand the motivations behind the model's prediction. In some domains, such as credit scoring and medical diagnostic, the lack of comprehensibility is a major issue and causes a reluctance to use the classifier or even complete rejection of the model. In a credit scoring context, when credit has been denied the Equal Credit Opportunity Act of the U.S. requires that the financial institution provides specific reasons why the customer's application was rejected, whereby vague reasons for denial are illegal. In the medical diagnostic domain as well, clarity and explainability are major constraints. The most suited classifiers for this type of problem are of course rules and trees. C4.5 is one of the techniques that constructs such comprehensible classifiers, but other techniques, such as rule extraction from neural network and support vector machine classifiers, have been proposed as well [4].

Our approach focuses on building accurate, though comprehensible classifiers, fit for dynamic, distributed environments.

2.2.3 Data Mining with Ant Systems

The first application of ant systems for data mining was reported in [28], where the authors introduce the AntMiner algorithm for the discovery of classification rules. Extensions and optimizations of the AntMiner are described in AntMiner2 and AntMiner3 [22]. The aim is to extract simple **if-then-else** rules from data, where the condition part of the rule is a conjunction of terms. All attributes are assumed to be categorical since the terms are of the form $< Variable = Value >$, e.g. $< Sex = male >$.

The original AntMiner works as follows. Each ant starts with an empty rule and chooses a term $< V_i = Value_k >$ to add to it's rule. The choice of the term to add is dependent on the pheromone function ($\tau(t)$) and the heuristic value (η) associated with each term. This choice is furthermore constrained since each variable can occur at most once in a rule to avoid inconsistencies such as $< Sex = male >$ **and** $< Sex = female >$. The ant keeps adding terms to its partial rule until either all variables have been used in the rule or if adding any term would make the rule cover less cases than a user-defined minimum. The class

predicted by this rule is determined by the majority class of the training cases covered by the rule. Afterwards the rule is pruned in order to remove irrelevant terms and the pheromone levels are adjusted, increasing the pheromone of the trail followed by the ant and evaporating the others. Another ant starts its search with the new pheromone trails to guide its search. This process is repeated until all ants have constructed a rule or when ants have already converged to the same constructed rule. The best rule among these constructed rules is added to the list of discovered rules and the training cases covered by this rule are removed from the training set. This overall process is repeated until the number of uncovered training cases is lower than a specific threshold.

The heuristic value in AntMiner is defined as an information theoretic measure in terms of the entropy, which can be seen as an impurity measure. AntMiner2 on the other hand uses a much simpler, though less accurate, density estimation equation as the heuristic value with the assumption that the small induced errors are compensated by the pheromone level. This makes AntMiner2 computationally less expensive without a degradation of the performance. Two key changes have been proposed in AntMiner3 [22], resulting in an increased accuracy. A different update rule is used and more exploration is incorporated with a different transition rule that increases the probability of choosing terms not yet used in previously constructed rules.

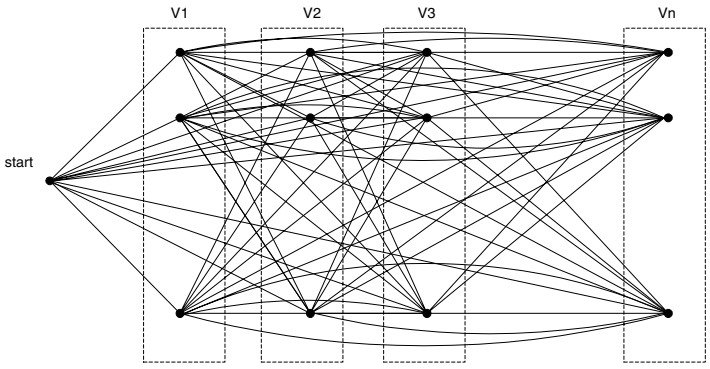


Fig. 2.1. Construction Graph of AntMiner

In these AntMiner versions, an ant can add terms corresponding to any of the variables that are not yet present in the partially constructed rule, with any of its values. This approach is illustrated in Fig. 2.1 which shows a graph representation of the ants' environment. Each 'column' or node group corresponds to a variable and every 'row' corresponds to a value. Each ant going to node $n_{i,k}$ (node in column i and row k) adds the term $< V_i = Value_k >$ to its rule. All ants begin in the start node and then start adding terms by walking through the construction graph representing the problem domain. As shown by Fig. 2.1, the complexity of the construction graph, measured by the number of edges, is $O(\frac{avg^2}{2} \cdot n^2)$ with n the number of variables and avg the average number of values per variable.

$$n \cdot avg + (n-1) \cdot avg^2 + (n-2) \cdot avg^2 + \ldots + avg^2 \approx avg^2 \cdot \frac{n(n+1)}{2} \tag{2.1}$$

2.3 AntMiner+

We build further on the work introduced in the previous AntMiner versions and try to resolve some issues. First of all, we define the environment as a directed acyclic construction graph which allows a clear representation of the problem domain and considerably improves the performance of the ant system. Furthermore, we introduce the better performing $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for mining rules. To the best of our knowledge, there is no use of the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system technique for the discovery of classification rules.

The main working of our algorithm is described in pseudo-code 2.3.1 below. First, a directed acyclic construction graph is created that acts as the environment of the ants. All ants begin in the start node and walk through their environment to the end node, gradually constructing a rule. Only the ant that describes the best rule, i.e. covers the most training points, will have the pheromone of its followed trail increased. Evaporation decreases the pheromone of all edges. Supplementary modifications of the pheromone levels may be needed since the $\mathcal{MAX}$-$\mathcal{MIN}$ approach additionally requires the pheromone levels to lie within a given interval. Since the probabilities are the same for all ants in the same iteration, these values are calculated in advance. When all the edges of one path have a pheromone level τ_{max} and all others edges have pheromone level τ_{min}, the rule corresponding to the path with τ_{max} will be extracted and training data covered by this rule removed from the training set. This iterative process will be repeated until enough training points have been covered or when early stopping occurs (cf. Sect. 2.3.5). Details of AntMiner+ are provided in the next sections.

Pseudo-code 2.3.1 AntMiner+

```
construct graph
while (not min. percentage of training data covered or early stopping)
  initialize heuristics, pheromones and probabilities of edges
  while (not converged)
      create ants
      let ants run from source to sink
      evaporize edges
      update path of best ant
      adjust pheromone levels if outside boundaries
      kill ants
      update probabilities of edges
  end
  extract rule
  flag the data points covered by the extracted rule
end
evaluate performance on test set
```

2.3.1 The Construction Graph

The AntMiner+ construction graph is defined as a simple DAG which provides a comprehensible view of the solution space. Ants in a node of variable V_i are only allowed to go to nodes of variable V_{i+1}. Consequently, each path from start to end node represents a rule. Similarly as before, each ant going from node $n_{i,j}$ to node $n_{i+1,k}$ adds the term $< V_{i+1} = Value_k >$ to its rule. Since binary classification is performed, at the end node the rule consequent $< class = 1 >$ is added to the rule. So during the walk from start to stop, an ant gradually constructs a complete rule. To allow for rules where not all variables are involved, hence shorter rules, an extra *dummy node* is added to each variable whose value is undetermined, meaning it can take any of the values available. This fits well in the construction graph and makes the need for pruning superfluous.

Although only categorical variables can be used in our implementation, we make a distinction between nominal and ordinal variables. Each nominal variable has one node group, but for the ordinal however, we build two node groups to allow for intervals to be chosen by the ants. The first node group corresponds to the lower bound of the interval and should thus be interpreted as $< V_{i+1} \geq Value_k >$, the second node group determines the upper bound, giving $< V_{i+2} \leq Value_l >$. This allows to have less, shorter and actually better rules. Note that in the ordinal case V_{i+1} is equal to V_{i+2}. Figure 2.2 gives a general view of the construction graph with the first variable being nominal and the second one ordinal, hence having two node groups. The complexity of this construction graph is $O(n \cdot avg^2)$, far below the complexity of the construction graph defined by previous

AntMiner versions[5]. The lower complexity of the AntMiner+ construction graph reduces the number of probability computations and makes the best possible term to add more obvious.

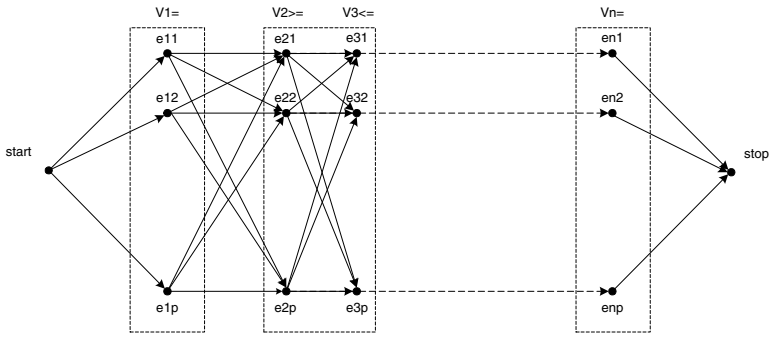


Fig. 2.2. Construction Graph of AntMiner+

2.3.2 Edge Probabilities

The probability $P_{(n_{i,j},n_{i+1,k})}$ is the probability that an ant which is in node $n_{i,j}$ (node for which variable V_i is equal to its j^{th} value) will go to node $n_{i+1,k}$. This probability is commonly defined as follows, with $|condition|$ denoting the number of datapoints fulfilling $condition$:

$$P_{(n_{i,j},n_{i+1,k})} = \frac{[\tau_{(n_{i,j},n_{i+1,k})}]^{\alpha}.[\eta_{n_{i+1,k}}]^{\beta}}{\sum_{l=1}^{|V_{i+1}|}[\tau_{(n_{i,j},n_{i+1,l})}]^{\alpha}.[\eta_{n_{i+1,l}}]^{\beta}} \tag{2.2}$$

Notice that this probability is dependent on two values: the heuristic value η and the pheromone value τ. The relative weights of these values are determined by the α and β parameters.

2.3.3 Heuristic Value

The heuristic value gives for each node in the construction graph a notion of its quality in the problem domain. For data mining, the quality is usually measured by the number of data points that are covered (described) by a value. Since we extract rules for the $class = 1$ case, we define the heuristic value for the node $n_{i,k}$ as follows:

[5]Note that n is now a value slightly higher than the number of variables since ordinal variables have two node groups.

$$\eta_{n_{i,k}} = \frac{|V_i = Value_k \ \& \ class = 1|}{|V_i = Value_k|} \tag{2.3}$$

As mentioned before, more complex and accurate formulas can be provided here, e.g. information entropy measures. This extra information however is computationally demanding and is already included (implicitly) in the pheromone value: small potential errors in the heuristic value will be compensated by the pheromone levels. The same argument goes for the fact that we do not take into account the history of the ant in the heuristic value, that is we do not look at how many data points are already covered by the rule so far. This implies that the sequence of the variables in the construction graph is irrelevant.

2.3.4 Pheromone Updating

Generally, updating the pheromone trail of an ant system is accomplished in two phases, being evaporation and reinforcement. Applying the ideas of $\mathcal{MAX}$-$\mathcal{MIN}$ ant systems has direct consequences for the pheromone updating rule in the following way:

Evaporation

Evaporation in an ant system is accomplished by diminishing the pheromone level of each trail according to the following rule:

$$\tau_{(n_{i,j},n_{i+1,k})}(t+1) = \rho \cdot \tau_{(n_{i,j},n_{i+1,k})}(t), \tag{2.4}$$

where ρ is the evaporation factor. Typical values for ρ lie in the range $[0.8, 0.99]$ [41].

Reinforcement

In a $\mathcal{MAX}$-$\mathcal{MIN}$ ant system reinforcement of the pheromone trail is only applied to the best ant's path. The best ant can be chosen as either the iteration best ant, or the global best ant. Results described in the literature bias our choice towards the iteration best ant [41]. This means that, taking into account the evaporation factor as well, the update rule for the best ant's path can be described by:

$$\tau_{(n_{i,j},n_{i+1,k})}(t+1) = \rho \cdot \tau_{(n_{i,j},n_{i+1,k})}(t) + \Delta^{best} \tag{2.5}$$

Clearly, the reinforcement of the best ant's path, Δ^{best}, should be proportional to the quality of the path. For data mining, we define the quality of a rule by the sum of its *confidence* and its *coverage*. Confidence is an indication of the number of correctly classified data points by a rule compared to the total number of data points covered by that rule. The coverage gives an indication of the overall importance of the specific rule by looking at the number of correctly classified

data points over the total number of uncovered data points. More formally, the pheromone amount to add to the path of the iteration best ant is given by the benefit of the path of the iteration best ant, as indicated by 2.6, with $rule_{ib-ant}$ being the rule described by the iteration best ant, and Cov a boolean expressing whether a datapoint is already covered by one of the extracted rules or not.

$$\Delta^{best} = \underbrace{\frac{|rule_{ib-ant}\ \&\ Class = 1|}{|rule_{ib-ant}|}}_{\textbf{confidence}} + \underbrace{\frac{|rule_{ib-ant}\ \&\ Class = 1|}{|Cov = 0|}}_{\textbf{coverage}} \tag{2.6}$$

Let us consider a dataset with 1000 data points and 2 ants extracting rules. Ant 1 describes rule R_1 that covers 100 data points, of which 90 are correctly described. Ant 2 extracts rule R_2, that covers 300 data points, of which 240 are correctly classified. As Table 2.2 shows rule R_2 has a higher quality and therefore ant 2 will be the iteration best ant.

Table 2.2. Example on calculation of rule quality

Rule	Confidence	Coverage	Quality
R_1	$\frac{90}{100} = 0.9$	$\frac{100}{1000} = 0.1$	1.0
R_2	$\frac{240}{300} = 0.8$	$\frac{300}{1000} = 0.3$	1.1

τ boundaries

An additional restriction imposed by the $\mathcal{MAX}$-$\mathcal{MIN}$ ant systems is that the pheromone level of the edges is restricted by an upper-bound (τ_{max}) and a lower-bound (τ_{min}). Furthermore, these bounds are dynamically altered during the execution of the ant system, so the values converge to the exact maximum value of τ_{max}. Every time an *iteration best* ant improves the results of the current *global best* ant, the boundaries should be updated in the following way [41]:

$$\tau_{max} = \frac{1}{1-\rho}.\Delta^{best} \tag{2.7}$$

The lower-bound (τ_{min}) for the pheromone level is derived from the probability that the best solution is found upon convergence p_{best}, the maximum pheromone level τ_{max} and the average number of values per variable. The exact derivation is given in [41].

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(avg-1) \cdot \sqrt[n]{p_{best}}} \tag{2.8}$$

2.3.5 Early Stopping

AntMiner+ stops constructing rules when either a predefined percentage of training points has been covered by the inferred rules or when early stopping occurs. The last criterion is often used in data mining and is explained in more detail.

The ultimate goal of AntMiner+ is to produce a model which performs well on new, unseen test instances. If this is the case, we say that the classifier generalises well. To do so, we basically have to avoid the classifier from fitting the noise or idiosyncrasies in the training data. This can be realized by monitoring the error on a separate validation set during the training session. When the error measure on the validation set starts to increase, training is stopped, thus effectively preventing the rule base from fitting the noise in the training data. This stopping technique is known as *early stopping*. Note that this causes loss of data that cannot be used for constructing the rules and hence, this method may not be appropriate for small data sets.

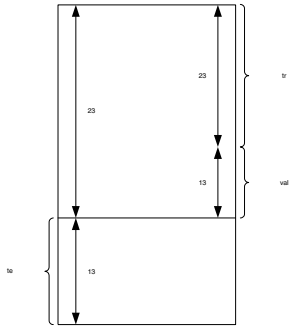


Fig. 2.3. Dataset splitup

The dataset is split up in three sets, as indicated in Fig. 2.3: a training set used by AntMiner+ to infer rules from, a validation set to implement the early stopping rule, and a test set to calculate an unbiased performance estimate. As a rule of thumb, two third of the complete dataset is typically used for training and validation and the remaining third for testing. Of the data points set aside for training and validation, two third is training data and one third validation data. A typical plot of the accuracy for the three types of data is shown in Fig. 2.4. The data used here is the german credit scoring dataset described further on in this chapter. As the figure shows, early stopping makes sure the rules generalise well. Going beyond the induced stop increases the training accuracy, yet lowers the validation (and also test) accuracy because overfitting occurs.

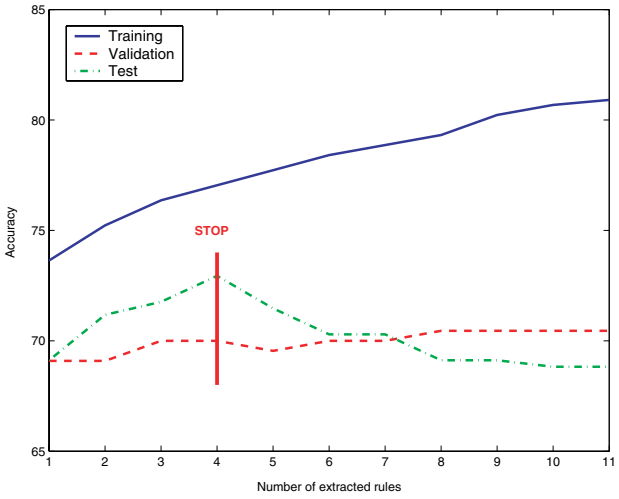


Fig. 2.4. Illustration of the early stopping rule

2.4 Distributed Data Mining With AntMiner+: a Credit Scoring Case

To demonstrate the possible use of AntMiner+ in a dynamic, distributed environment we developed the following credit scoring case, see Fig. 2.5. This case has not yet been put in practice and should only be viewed as an illustration of how to distribute AntMiner+.

One of the key decisions financial institutions have to make is to decide whether or not to grant a loan to an applicant. This basically comes down to a binary classification problem which aims at distinguishing the good payers from the bad payers. Suppose a bank wants to make use of AntMiner+ to build comprehensible classifiers on which to base its credit approval decisions. The bank has some customer data at its disposal such as amount on checking and savings account, possibly distributed over several branch offices. It also has access to data from a credit bureau (such as Experian or Equifax) which consolidates information from a variety of financial institutions. Further available data comes from the Tax and Housing Offices. The externally provided data is continuously updated. A centralized data mining approach would require continuous importing of data across the network, a rather cumbersome task. AntMiner+ on the other hand, sends the ants through the network to the relevant databases. Now the construction graph is dynamic and distributed over different sites, where each database contains a number of node groups. In practice, of course, support for the use of AntMiner+, common customer IDs and routing knowledge needs to be incorporated.

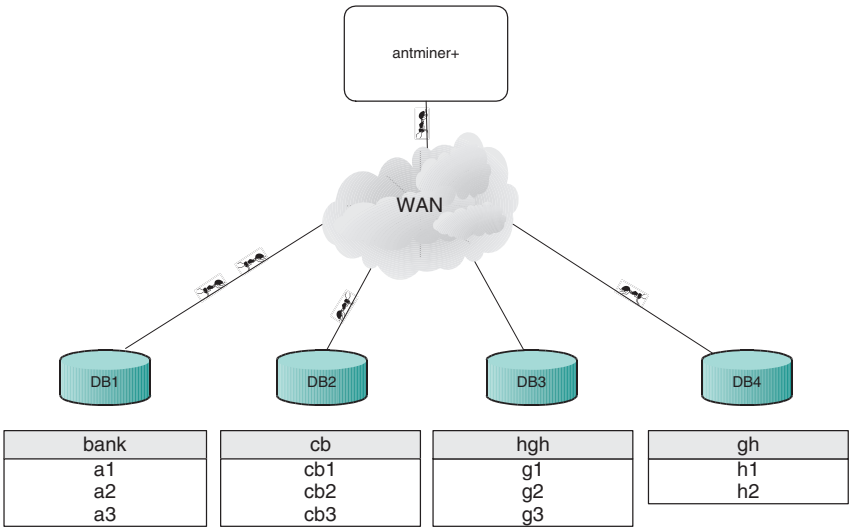


Fig. 2.5. AntMiner+ in distributed environment

The decentralized approach described here which operates in a dynamic, distributed environment allows to detect new customer default profiles quickly, is easily scalable, and robust to failure: ants going lost over the network, or even losing the connection to databases do not severely degrade the performance and AntMiner+ will still be able to work properly.

2.5 Experiments and Results

2.5.1 Experimental Set-Up

We applied AntMiner+ to several publicly available datasets. Training, validation and test set are determined in the manner described before. To eliminate any chance of having unusually good or bad training and test sets, 10 runs are conducted where the data is first randomized before the training, validation and test set are chosen. Experiments are conducted with 1000 ants, the evaporation rate ρ set at 0.85, and the relative weights α and β set at respectively 2 and 1.

A wide range of techniques are chosen to benchmark the AntMiner+ classifier against. C4.5 is the popular decision tree builder [34] where each leaf assigns class labels to observations. Each of these leaves can be represented by a rule; therefore C4.5 is also a truly comprehensible classifier. K-nearest neighbor

classifiers (kNN) classify a data instance by considering only the k most similar data points in the training set. We apply kNN to our datasets for k = 1 and k = 10. This technique is called lazy since it does not involve creating a classification model, but rather defers the decisions on how to classify new data points beyond the training session. For both C4.5 and kNN we use the Weka workbench [34]. Also included is the commonly used logistic regression (logit) classifier and the non-linear support vector machine (SVM) classifier [44]. We take the Least Squares SVM variant [43] with RBF kernel using the LS-SVM Matlab toolbox [42].

2.5.2 Datasets

Different datasets are chosen to infer rules from by AntMiner+. Two datasets concern medical diagnosis of breast cancer and one dataset concerns credit scoring; these datasets are chosen for the critical importance of comprehensibility of the generated classifiers. Other datasets used concern the tic-tac-toe toy problem and the two-dimensional ripley dataset which allows for visualization of the extracted rules. Both the tic-tac-toe, the credit scoring and the breast cancer datasets come from the publicly available UCI data repository [21].

Breast Cancer Diagnosis

The task at hand for the breast cancer diagnosis datasets consists of classifying breast masses as being either benign or malignant. For this, attributes of a sample are listed that are deemed relevant. Two different datasets are used, one obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [23] and the other from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia and is provided by M. Zwitter and M. Soklic [26]. This is one of the domains where comprehensibility is a key issue and thus rules are very much preferred. The following rule, with 97.6% training accuracy and 95.7% test accuracy, was extracted by AntMiner+ for the Breast Cancer Wisconsin dataset:

Table 2.3. Example rule on Breast Cancer Wisconsin dataset

if (Clump Thickness $\in [1,8]$ **and** Uniformity of Cell Size $\in [1,9]$ **and**
Uniformity of Cell Shape $\in [1,8]$ **and** Marginal Adhesion $\in [1,6]$ **and**
Single Epithelial Cell Size $\in [1,9]$ **and** Bare Nuclei $\in [1,5]$ **and**
Bland Chromatin $\in [1,8]$ **and** Normal Nucleoli $\in [1,9]$ **and** Mitoses $\in [1,2]$)
then class = benign
else class = malignant

Credit Scoring

As mentioned before, credit scoring involves the discrimination between good payers and bad ones. We used the german credit scoring dataset offered by prof.

Hofmann. An AntMiner+ example rule set is described in Table 2.4 which has a training accuracy of 75.5% and a test accuracy of 72.1%.

Table 2.4. Example rule set on German Credit Scoring Dataset

if (Checking Account $<$ 200 DM **and** Duration $>$ 15 m **and**
Credit History = no credits taken **and** Savings Account $<$ 1000 DM)
then class = bad
else if (Purpose = new car/repairs/education/others **and**
Credit History = no credits taken/all credits paid back duly at this bank **and**
Savings Account $<$ 1000 DM)
then class = bad
else if (Checking Account $<$ 0 DM **and**
Purpose = furniture/domestic appliances/business **and**
Credit History = no credits taken/all credits paid back duly at this bank **and**
Savings Account $<$ 500 DM)
then class = bad
else if (Checking Account $<$ 0 DM **and** Duration $>$ 15 m **and**
Credit History = delay in paying off in the past **and**
Savings Account $<$ 500 DM)
then class = bad
else class = good

Toy Problems

Also included in our experiments is the *tic-tac-toe* dataset, which encodes the complete set of possible board configurations at the end of tic-tac-toe games where X is assumed to have played first. The target concept is 'win for X' (i.e., true when X has one of 8 possible ways to create a 'three-in-a-row'). The extracted rules can easily be verified in Table 2.5 that shows the board of the game and the 9 variables which can take value X, O or B (blank). An example of a rule base extracted by AntMiner+ is provided in Table 2.6.

Table 2.5. tic-tac-toe game

A1	A2	A3
A4	A5	A6
A7	A8	A9

Table 2.6. Example rule on tic-tac-toe dataset

if (A7 = 1 **and** A8 = 1 **and** A9 = 1) **then** class = X
else if (A3 = 1 **and** A5 = 1 **and** A7 = 1) **then** class = X
else if (A1 = 1 **and** A5 = 1 **and** A9 = 1) **then** class = X
else if (A3 = 1 **and** A6 = 1 **and** A9 = 1) **then** class = X
else if (A4 = 1 **and** A5 = 1 **and** A6 = 1) **then** class = X
else if (A2 = 1 **and** A5 = 1 **and** A8 = 1) **then** class = X
else if (A1 = 1 **and** A4 = 1 **and** A7 = 1) **then** class = X
else if (A1 = 1 **and** A2 = 1 **and** A3 = 1) **then** class = X
else class = O

Ripley's dataset [32] has two variables and two classes, where the classes are drawn from two normal distributions with a high degree of overlap. This two-dimensional dataset allows for visualization of the classifiers. Since AntMiner+ can only deal with categorical variables, the continuous values of the two variables are divided into 50 intervals of equal length. The Ripley dataset is shown in Fig. 2.6, together with the decision boundary defined by the rules extracted by AntMiner+ (accuracy 90.8%), and a support vector machine model (accuracy 91.4%).

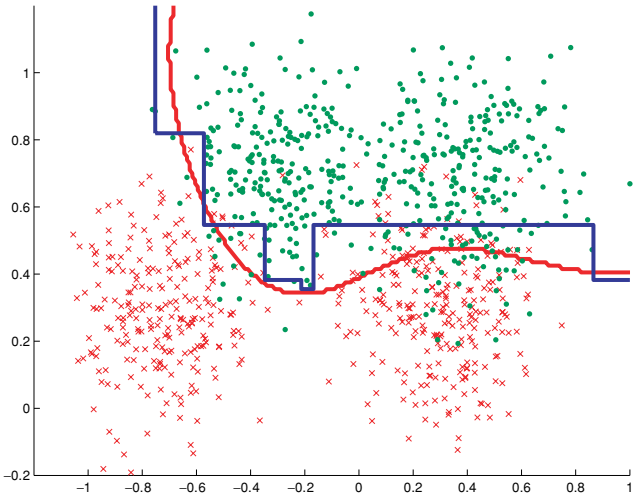
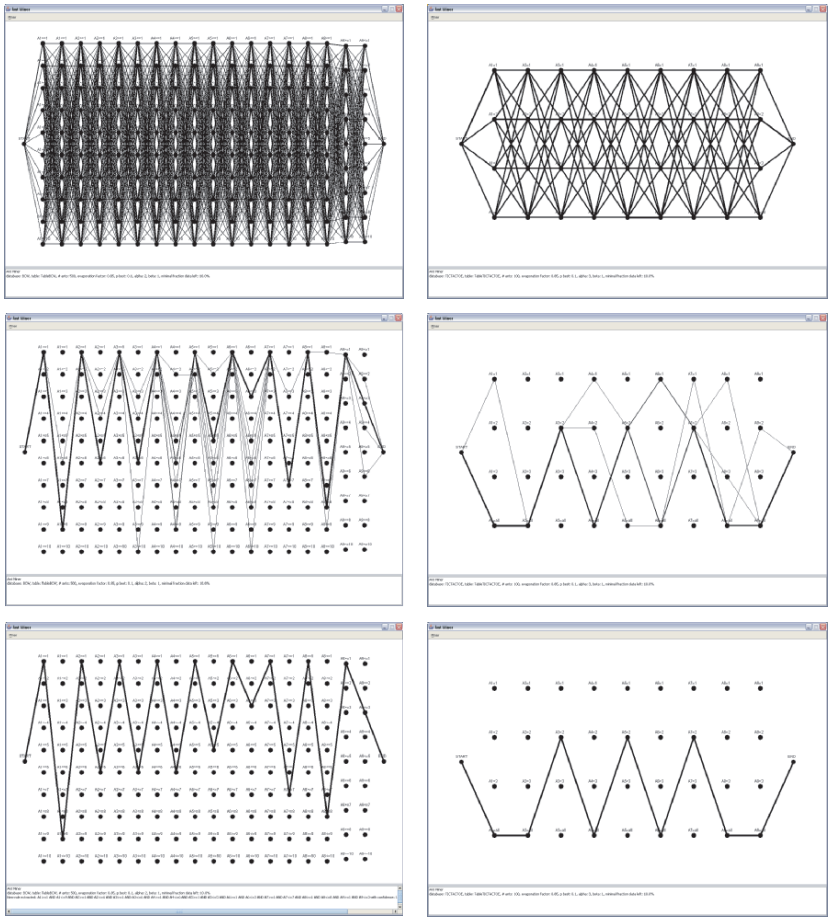


Fig. 2.6. AntMiner rules (■) and SVM decision boundary (■) for Ripley's dataset

2.5.3 Software Implementation

AntMiner+ is implemented in the platform-independent, object-oriented Java programming environment, with usage of the MySQL open source database server. Several screenshots of the Graphical User Interface (GUI) of AntMiner+, applied to the breast cancer wisconsin and tic-tac-toe datasets, are provided in Fig. 2.7. The GUI shows the construction graph with the width of the edges being proportional to their pheromone value. Extracted rules, with their training, validation and test accuracy, are displayed in the bottom box.



(a) BCW Dataset (b) tic-tac-toe Dataset

Fig. 2.7. Screenshots of AntMiner+ during different stages of execution: from initialization (top) to convergence (bottom)

2.5.4 Discussion

The results of our experiments are shown in Table 2.7. For each dataset, the number of data instances (inst) and attributes (attr) as well as the accuracy and number of generated rules are displayed. The techniques are ranked according to their accuracy and the average ranking (AR) over the different datasets of each technique is included in the table, hence a low AR indicates good performance. The best average test set performance over 10 runs is underlined and denoted in bold face for each data set. We then use a paired t-test to test the performance differences. Performances that are not significantly different at the 5% level from the top performance with respect to a one-tailed paired t-test are tabulated in bold face. Statistically significant underperformances at the 1% level are emphasized in italics. Performances significantly different at the 5% level but not at the 1% level are reported in normal script. Results published for AntMiner1 and 3, reported in [29, 22], are also listed in the table but not included in the comparison as just described since we did not conduct experiments with these AntMiner versions ourselves.

Table 2.7. Average out-of-sample performance

		bcw		**bcl**		**ger**		**ttt**		**rip**	
		inst	attr	inst	attr	inst	attr	inst	attr	inst	attr
		683	9	277	9	1000	19	958	9	1250	2
Technique	**AR**	Acc	#R	Acc	#R	Acc	#R	Acc	#R	Acc	#R
AntMiner		92.63	10.1	75.28	7.1			70.99	16.5		
AntMiner3		94.32	13.2					76.58	18.6		
AntMiner+	3.2	95.79	1	**77.05**	3.9	*72.29*	3.9	**99.76**	8	*89.41*	3.9
C4.5	4.6	**94.38**	11	75.68	21	**72.91**	36	*84.17*	95	*89.08*	6
1NN	4.4	**95.84**		*74.74*		*72.48*		*97.98*		*88.66*	
10NN	1.8	**96.48**		**78.42**		**74.26**		*95.18*		**90.80**	
SVM	3.8	**92.81**		**76.56**		**73.68**		91.06		**89.78**	
logit	3.2	**96.54**		**76.77**		**75.24**		*65.56*		*88.92*	

The best performance is achieved by 10NN with an average ranking of 1.8. However, the nearest neighbor techniques are lazy in the sense that there is no actual classifier. Comprehensibility of such decisions, based on the similarity with training data, is limited. The SVM models perform consistently well, but the non-linear, complex nature of the generated classifiers makes them rather incomprehensible for humans. Logistic regression achieves good results as well but is troubled with similar opacity issues. Equations 2.9 and 2.10 describe the form of respectively the SVM and logistic regression classifiers and clearly indicate the opacity of these models.

$$y_{SVM}(\mathbf{x}) = \text{sign}[\sum_{i=1}^{N} \alpha_i y_i \exp\{-\|\mathbf{x}-\mathbf{x}_i\|_2^2/\sigma^2\} + b] \tag{2.9}$$

$$y_{logit}(\mathbf{x}) = 1/(1+\exp\{-(w_0+\mathbf{w}^T\mathbf{x})\}) \tag{2.10}$$

The only techniques that deal with the comprehensibility aspect are C4.5 and Antminer+. With an overall average ranking of 3.2 AntMiner+ holds a top three place among the included state-of-the-art classification techniques. AntMiner+ outperforms C4.5 on all but one dataset and consistently does so with fewer rules, making AntMiner+ the best performing technique when considering both accuracy and comprehensibility.

The better results can be attributed to our $\mathcal{MAX}$-$\mathcal{MIN}$ approach, our simple construction graph with the inclusion of dummy nodes, as well as our ability to include intervals in our rules. The $\mathcal{MAX}$-$\mathcal{MIN}$ ant system is better able to combine exploration of the search space and exploitation of the best solutions found, and has been shown to perform better than the usual ant system [9, 37, 41]. The construction graph modeled as a DAG reduces the complexity of the problem, yet the presence of dummy nodes enables AntMiner+ to infer short rules. Surely, the intervals play a crucial role as well in attaining fewer and shorter rules. This is best demonstrated with the breast cancer wisconsin dataset. The only possible way to achieve an accurate classifier with only 1 rule is when intervals are allowed. Several weaknesses need to be kept in mind as well. AntMiner+ requires more computational time than C4.5 to achieve its results[6] and is only able to deal with categorical variables. Parallelization of the inherently distributed AntMiner+ system could decrease the computation time needed.

2.6 Conclusion and Future Research

AntMiner+ is a technique that successfully incorporates swarm intelligence in data mining. Using a $\mathcal{MAX}$-$\mathcal{MIN}$ system, AntMiner+ builds comprehensible, accurate rule-based classifiers that perform competitively with state-of-the-art classification techniques. Although ants have a limited memory and perform actions based on local information only, the ants come to complex behavior due to self-organization and indirect communication. The intrinsic properties of ant systems allow us to easily distribute our approach to provide robust and scalable data mining solutions.

Still, several challenges lie ahead. Most real-life datasets contain various continuous variables. One approach to deal with this is by categorizing these variables in a pre-processing step. Incorporating the variables in a dynamically

[6] Time order of minutes up to one hour on Xeon 2.4GHz, 1GB RAM for the datasets analyzed.

changing construction graph is another option and focus of current research. An issue faced by ant systems, is the necessity to instantiate various parameters, such as the weight parameters α and β. These parameters are typically determined by trial and error or with more fancy techniques such as genetic algorithms or local search techniques. We are currently investigating the possibility of including these two variables in the construction graph, which would have the supplementary benefit of generating dynamically changing parameters as the environment changes. Once again, the ants will take over the work from the user.

Acknowledgment

We would like to thank the Flemish Research Council (FWO, Grant G.0615.05), and the Microsoft and KBC-Vlekho-K.U.Leuven Research Chairs for financial support to the authors.

References

1. Abraham A, Ramos V (2003) Web usage mining using artificial ant colony clustering. In Proceedings of Congress on Evolutionary Computation (CEC2003), Australia, IEEE Press, ISBN 0780378040, 1384-1391
2. Baesens B (2003) Developing intelligent systems for credit scoring using machine learning techniques. PhD thesis, K.U.Leuven
3. Baesens B, Van Gestel T, Viaene S, Stepanova M, Suykens J, Vanthienen J (2003) Benchmarking state-of-the-art classification algorithms for credit scoring. Journal of the Operational Research Society, 54(6):627–635
4. Baesens B, Setiono R, Mues C, Vanthienen J (2003) Using neural network rule extraction and decision tables for credit-risk evaluation. Management Science, 49(3):312–329
5. Bonabeau E, Dorigo M, Theraulaz G (2001) Swarm intelligence: From natural to artificial systems. Journal of Artificial Societies and Social Simulation, 4(1)
6. Bullnheimer B, Hartl RF, Strauss C (1999) Applying the ant system to the vehicle routing problem. In: Osman IH, Roucairol C, Voss S, Martello S (eds) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization
7. Di Caro G, Dorigo M (1998) Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 9:317–365
8. Cicirello VA, Smith SF (2001) Ant colony control for autonomous decentralized shop floor routing. In: the Fifth International Symposium on Autonomous Decentralized Systems, pages 383–390
9. Colorni A, Dorigo M, Maniezzo V, Trubian M (1994) Ant system for jobshop scheduling. Journal of Operations Research, Statistics and Computer Science, 34(1):39–53
10. Dorigo M Ant colony optimization [http://iridia.ulb.ac.be/ mdorigo/aco/aco.html].

11. Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy. Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, IT
12. Dorigo M, Maniezzo V, Colorni A (1996) The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 26(1):29–41
13. Eyckelhof CJ, Snoek M (2002) Ant systems for a dynamic tsp. In: ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms, pages 88–99, London, UK. Springer-Verlag
14. Forsyth P, Wren A (1997) An ant system for bus driver scheduling. Research Report 97.25, University of Leeds School of Computer Studies
15. Gambardella LM, Taillard E, Dorigo M (1999) Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society, (50):167–176
16. Gambardella LM, Dorigo M (1995) Ant-q: A reinforcement learning approach to the traveling salesman problem. In: Proceedings of the Eleventh International Conference on Machine Learning, pages 252–260
17. Gambardella LM, Dorigo M (1996) Solving symmetric and asymmetric tsps by ant colonies. In: Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96), pages 622–627
18. Grassé PP (1959) La reconstruction du nid et les coordination inter-individuelles chez bellicositermes natalensis et cubitermes sp. la thérie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insect. Soc., 6:41–80
19. Hand D (2002) Pattern detection and discovery. In: Hand D, Adams N, Bolton R (eds) Pattern Detection and Discovery, volume 2447 of Lecture Notes in Computer Science, pages 1–12. Springer
20. Handl J, Knowles J, Dorigo M (2003) Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som. Technical Report TR/IRIDIA/2003-24, Universite Libre de Bruxelles
21. Hettich S, Bay SD (1996) The uci kdd archive [http://kdd.ics.uci.edu]
22. Liu B, Abbass HA, McKay B (2004) Classification rule discovery with ant colony optimization. IEEE Computational Intelligence Bulletin, 3(1):31–35
23. Mangasarian OL, Wolberg WH (1990) Cancer diagnosis via linear programming. SIAM News, 23(5):1–18
24. Maniezzo V (1998) Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Research CSR 98-1, Scienze dell'Informazione, Università di Bologna, Sede di Cesena, Italy
25. Maniezzo V, Colorni A (1999) The ant system applied to the quadratic assignment problem. IEEE Transactions on Knowledge and Data Engineering
26. Michalski RS, Mozetic I, Hong J, Lavrac N (1986) The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In: AAAI, pages 1041–1047
27. Naisbitt J (1988) Megatrends : Ten New Directions Transforming Our Lives. Warner Books
28. Parpinelli RS, Lopes HS, Freitas AA (2001) An ant colony based system for data mining: Applications to medical data. In: Lee Spector, Goodman E, Wu A, Langdon WB, Voigt H, Gen M, Sen S, Dorigo M, Pezeshk S, Garzon M, Burke E (eds) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pages 791–797, San Francisco, California, USA, 7-11. Morgan Kaufmann

29. Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation, 6(4):321–332
30. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
31. Ramos V, Abraham A (2003) Swarms on continuous data. In: Proceedings of the Congress on Evolutionary Computation, IEEE Press, pages 1370–1375
32. Ripley BD (1994) Neural networks and related methods for classification. Journal of the Royal Statistical Society B, 56:409–456
33. Schockaert S, De Cock M, Cornelis C, Kerre EE (2004) Efficient clustering with fuzzy ants. Applied Computational Intelligence
34. Schoonderwoerd R, Holland OE, Bruten JL, Rothkrantz LJM (1996) Ant-based load balancing in telecommunications networks. Adaptive Behavior, (2):169–207
35. Socha K, Knowles J, Sampels M (2002) A $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for the university timetabling problem. In: Dorigo M, Di Caro G, Sampels M (eds) Proceedings of ANTS 2002 – Third International Workshop on Ant Algorithms, volume 2463 of Lecture Notes in Computer Science, pages 1–13. Springer-Verlag, Berlin, Germany
36. StützleT, Dorigo M (1999) Aco algorithms for the quadratic assignment problem. In: Dorigo M, Corne D, Glover F (eds) New Ideas in Optimization
37. Stützle T, Hoos HH (1996) Improving the ant-system: A detailed report on the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. Technical Report AIDA 96-12, FG Intellektik, TU Darmstadt, Germany
38. Stützle T, Hoos HH (1997) The $\mathcal{MAX}$-$\mathcal{MIN}$ ant system and local search for the traveling salesman problem. In: Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), pages 309–314
39. Stützle T, Hoos HH (1998) Improvements on the ant system: Introducing the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. In: Steele NC, Albrecht RF, Smith GD (eds) Artificial Neural Networks and Genetic Algorithms, pages 245–249
40. Stützle T, Hoos HH (1999) $\mathcal{MAX}$-$\mathcal{MIN}$ ant system and local search for combinatorial optimization problems. In: Osman IH, Voss S, Martello S, Roucairol C (eds) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pages 313–329
41. Stützle, Hoos HH (2000) $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. Future Generation Computer Systems, 16(8):889–914
42. Suykens JAK, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J (2002) Least Squares Support Vector Machines. World Scientific, Singapore
43. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process. Lett., 9(3):293–300
44. Vapnik VN (1995) The nature of statistical learning theory. Springer-Verlag, New York, NY, USA
45. Wade A, Salhi S (2004) An ant system algorithm for the mixed vehicle routing problem with backhauls. In: Metaheuristics: computer decision-making, pages 699–719, Norwell, MA, USA, 2004. Kluwer Academic Publishers
46. Witten IH, Frank E (2000) Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA